Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME

Vincent Roca, Mathieu Cunche, Jérôme Lacan, Amine Bouabdallah, Kazuhisa Matsuzono

Internet Engineering Task Force (IETF)
Request for Comments: 6865
Category: Standards Track
ISSN: 2070-1721

V. Roca
INRIA
M. Cunche
INSA-Lyon/INRIA
J. Lacan
ISAE, Univ. of Toulouse
A. Bouabdallah
CDTA
K. Matsuzono
Keio University

February 2013

Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME

Abstract

This document describes a fully-specified simple Forward Error Correction (FEC) scheme for Reed-Solomon codes over the finite field (also known as the Galois Field) GF(2^^m), with 2 <= m <= 16, that can be used to protect arbitrary media streams along the lines defined by FECFRAME. The Reed-Solomon codes considered have attractive properties, since they offer optimal protection against packet erasures and the source symbols are part of the encoding symbols, which can greatly simplify decoding. However, the price to pay is a limit on the maximum source block size, on the maximum number of encoding symbols, and a computational complexity higher than that of the Low-Density Parity Check (LDPC) codes, for instance.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6865.

Table of Contents

1. Introduction

The use of the Forward Error Correction (FEC) codes is a classic solution to improve the reliability of unicast, multicast, and broadcast Content Delivery Protocols (CDP) and applications. [RFC6363] describes a generic framework to use FEC schemes with media delivery applications, and for instance with real-time streaming media applications based on the Real-time Transport Protocol (RTP). Similarly, [RFC5052] describes a generic framework to use FEC schemes with object delivery applications (where the objects are files, for example) based on the Asynchronous Layered Coding (ALC) [RFC5775] and NACK-Oriented Reliable Multicast (NORM) [RFC5740] transport protocols.

More specifically, the [RFC5053] and [RFC5170] FEC schemes introduce erasure codes based on sparse parity-check matrices for object delivery protocols like ALC and NORM. These codes are efficient in terms of processing but not optimal in terms of erasure recovery capabilities when dealing with "small" objects.

The Reed-Solomon FEC codes described in this document belong to the class of Maximum Distance Separable (MDS) codes that are optimal in terms of erasure recovery capability. It means that a receiver can recover the k source symbols from any set of exactly k encoding symbols. These codes are also systematic codes, which means that the k source symbols are part of the encoding symbols. However, they are limited in terms of maximum source block size and number of encoding symbols. Since the real-time constraints of media delivery applications usually limit the maximum source block size, this is not considered to be a major issue in the context of FECFRAME for many (but not necessarily all) use cases. Additionally, if the encoding/decoding complexity is higher with Reed-Solomon codes than it is with [RFC5053] or [RFC5170] codes, it remains reasonable for most use cases, even in case of a software codec.

Many applications dealing with reliable content transmission or content storage already rely on packet-based Reed-Solomon erasure recovery codes. In particular, many of them use the Reed-Solomon codec of Luigi Rizzo [RS-codec] [Rizzo97]. The goal of the present document is to specify a simple Reed-Solomon scheme that is compatible with this codec.

More specifically, [RFC5510] introduced such Reed-Solomon codes and several associated FEC schemes that are compatible with the [RFC5052] framework. The present document inherits from Section 8 of [RFC5510], "Reed-Solomon Codes Specification for the Erasure

Channel", the specifications of the core Reed-Solomon codes based on Vandermonde matrices and specifies a simple FEC scheme that is compatible with FECFRAME [RFC6363]:

> The Fully-Specified FEC Scheme with FEC Encoding ID 8 specifies a simple way of using of Reed-Solomon codes over GF(2^^m), with 2 <= m <= 16, in order to protect arbitrary Application Data Unit (ADU) flows.

Therefore, Sections 4, 5, 6, and 7 of [RFC5510] that define [RFC5052]-specific Formats and Procedures are not considered and are replaced by FECFRAME-specific Formats and Procedures.

For instance, with this scheme, a set of Application Data Units (ADUs) coming from one or several media delivery applications (e.g., a set of RTP packets), are grouped in an ADU block and FEC encoded as a whole. With Reed-Solomon codes over GF(2^^8), there is a strict limit over the number of ADUs that can be protected together, since the number of encoded symbols, n, must be inferior or equal to 255. This constraint is relaxed when using a higher finite field size (m > 8), at the price of an increased computational complexity.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Definitions Notations and Abbreviations

3.1. Definitions

This document uses the following terms and definitions. Some of these terms and definitions are FEC scheme specific and are in line with [RFC5052]:

Source symbol: unit of data used during the encoding process. In this specification, there is always one source symbol per ADU.

Encoding symbol: unit of data generated by the encoding process. With systematic codes, source symbols are part of the encoding symbols.

Repair symbol: encoding symbol that is not a source symbol.

Code rate: the k/n ratio, i.e., the ratio between the number of source symbols and the number of encoding symbols. By definition, the code rate is such that: 0 < code rate <= 1. A code rate close to 1 indicates that a small number of repair symbols have been produced during the encoding process.

Systematic code: FEC code in which the source symbols are part of the encoding symbols. The Reed-Solomon codes introduced in this document are systematic.

Source Block: a block of k source symbols that are considered together for the encoding.

Packet erasure channel: a communication path where packets are either dropped (e.g., by a congested router, or because the number of transmission errors exceeds the correction capabilities of the physical layer codes) or received. When a packet is received, it is assumed that this packet is not corrupted.

Some of these terms and definitions are FECFRAME specific and are in line with [RFC6363]:

Application Data Unit (ADU): The unit of source data provided as payload to the transport layer. Depending on the use case, an ADU may use an RTP encapsulation.

(Source) ADU Flow: A sequence of ADUs associated with a transport-layer flow identifier (such as the standard 5-tuple {Source IP address, source port, destination IP address, destination port, transport protocol}). Depending on the use case, several ADU flows may be protected together by FECFRAME.

ADU Block: a set of ADUs that are considered together by the FECFRAME instance for the purpose of the FEC scheme. Along with the flow ID (F[]), length (L[]), and padding (Pad[]) fields, they form the set of source symbols over which FEC encoding will be performed.

ADU Information (ADUI): a unit of data constituted by the ADU and the associated Flow ID, Length and Padding fields (Section 4.3). This is the unit of data that is used as source symbol.

FEC Framework Configuration Information (FFCI): Information that controls the operation of FECFRAME. The FFCI enables the synchronization of the FECFRAME sender and receiver instances.

FEC Source Packet: At a sender (respectively, at a receiver) a payload submitted to (respectively, received from) the transport protocol containing an ADU along with an Explicit Source FEC Payload ID (that must be present in the FEC scheme defined by the present document, see Section 5.1.2).

FEC Repair Packet: At a sender (respectively, at a receiver) a payload submitted to (respectively, received from) the transport protocol containing one repair symbol along with a Repair FEC Payload ID and possibly an RTP header.

The above terminology is illustrated in Figure 1 (sender's point of view):

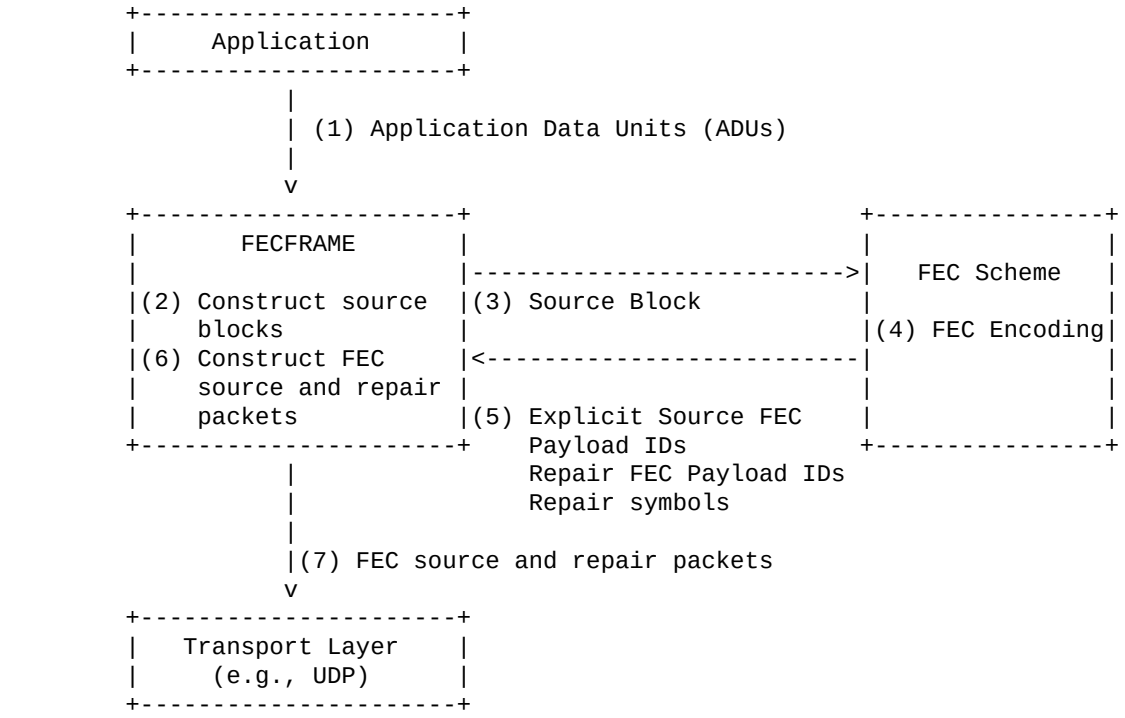

Figure 1: Terminology used in this document (sender).

3.2. Notations

This document uses the following notations. Some of them are FEC scheme specific.

k denotes the number of source symbols in a source block.

max_k denotes the maximum number of source symbols for any source block.

n denotes the number of encoding symbols generated for a source block.

E denotes the encoding symbol length in bytes.

GF(q) denotes a finite field (also known as the Galois Field) with q elements. We assume that q = 2^^m in this document.

m defines the length of the elements in the finite field, in bits. In this document, m is such that 2 <= m <= 16.

q defines the number of elements in the finite field. We have: q = 2^^m in this specification.

CR denotes the "code rate", i.e., the k/n ratio.

a^^b denotes a raised to the power b.

Some of them are FECFRAME specific:

B denotes the number of ADUs per ADU block.

max_B denotes the maximum number of ADUs for any ADU block.

3.3. Abbreviations

This document uses the following abbreviations:

ADU stands for Application Data Unit.

ADUI stands for Application Data Unit Information.

ESI stands for Encoding Symbol ID.

FEC stands for Forward Error (or Erasure) Correction code.

FFCI stands for FEC Framework Configuration Information.

FSSI stands for FEC Scheme-Specific Information.

MDS stands for Maximum Distance Separable code.

SBN stands for Source Block Number.

SDP stands for Session Description Protocol.

4. Common Procedures Related to the ADU Block and Source Block Creation

This section introduces the procedures that are used during the ADU block and the related source block creation for the FEC scheme considered.

4.1. Restrictions

This specification has the following restrictions:

- there MUST be exactly one source symbol per ADUI, and therefore per ADU;
- there MUST be exactly one repair symbol per FEC Repair Packet;
- there MUST be exactly one source block per ADU block.

4.2. ADU Block Creation

Two kinds of limitations exist that impact the ADU block creation:

- at the FEC Scheme level: the finite field size (m parameter) directly impacts the maximum source block size and the maximum number of encoding symbols;
- at the FECFRAME instance level: the target use case can have real-time constraints that can/will define a maximum ADU block size.

Note that terms "maximum source block size" and "maximum ADU block size" depend on the point of view that is adopted (FEC Scheme versus FECFRAME instance). However, in this document, both refer to the same value since Section 4.1 requires there is exactly one source symbol per ADU. We now detail each of these aspects.

The finite field size parameter m defines the number of non-zero elements in this field, which is equal to: q - 1 = 2^^m - 1. This q - 1 value is also the theoretical maximum number of encoding symbols that can be produced for a source block. For instance, when m = 8 (default) there is a maximum of 2^^8 - 1 = 255 encoding symbols. So: k < n <= 255. Given the target FEC code rate (e.g., provided by the end-user or upper application when starting the FECFRAME instance, and taking into account the known or estimated packet loss rate), the sender calculates:

```
max_k = floor((2^^m - 1) * CR)
```

This max_k value leaves enough room for the sender to produce the desired number of repair symbols. Since there is one source symbol per ADU, max_k is also an upper bound to the maximum number of ADUs per ADU block.

The source ADU flows can have real-time constraints. When there are multiple flows, with different real-time constraints, let us consider the most stringent constraints (see [RFC6363], Section 10.2, item 6 for recommendations when several flows are globally protected). In that case, the maximum number of ADUs of an ADU block must not exceed a certain threshold since it directly impacts the decoding delay. The larger the ADU block size, the longer a decoder may have to wait until it has received a sufficient number of encoding symbols for decoding to succeed, and therefore the larger the decoding delay. When the target use case is known, these real-time constraints result in an upper bound to the ADU block size, max_rt.

For instance, if the use case specifies a maximum decoding latency l, and if each source ADU covers a duration d of a continuous media (we assume here the simple case of a constant bit-rate ADU flow), then the ADU block size must not exceed:

```
max_rt = floor(l / d)
```

After encoding, this block will produce a set of at most n = max_rt / CR encoding symbols. These n encoding symbols will have to be sent at a rate of n / l packets per second. For instance, with d = 10 ms, l = 1 s, max_rt = 100 ADUs.

If we take into account all these constraints, we find:

```
max_B = min(max_k, max_rt)
```

This max_B parameter is an upper bound to the number of ADUs that can constitute an ADU block.

4.3. Source Block Creation

In their most general form, FECFRAME and the Reed-Solomon FEC scheme are meant to protect a set of independent flows. Since the flows have no relationship to one another, the ADU size of each flow can potentially vary significantly. Even in the special case of a single flow, the ADU sizes can largely vary (e.g., the various frames of a "Group of Pictures" (GOP) of an H.264 flow will have different sizes). This diversity must be addressed since the Reed-Solomon FEC scheme requires a constant encoding symbol size (E parameter) per

source block. Since this specification requires that there is only one source symbol per ADU, E must be large enough to contain all the ADUs of an ADU block along with their prepended 3 bytes (see below).

In situations where E is determined per source block (default, specified by the FFCI/FSSI with S = 0, Section 5.1.1.2), E is equal to the size of the largest ADU of this source block plus 3 (for the prepended 3 bytes; see below). In this case, upon receiving the first FEC Repair Packet for this source block, since this packet MUST contain a single repair symbol (Section 5.1.3), a receiver determines the E parameter used for this source block.

In situations where E is fixed (specified by the FFCI/FSSI with S = 1, Section 5.1.1.2), then E must be greater or equal to the size of the largest ADU of this source block plus 3 (for the prepended 3 bytes; see below). If this is not the case, an error is returned. How to handle this error is use-case specific (e.g., a larger E parameter may be communicated to the receivers in an updated FFCI message using an appropriate mechanism) and is not considered by this specification.

The ADU block is always encoded as a single source block. There are a total of B <= max_B ADUs in this ADU block. For the ADU i, with 0 <= i <= B-1, 3 bytes are prepended (Figure 2):

- The first byte, F[i] (Flow ID), contains the integer identifier associated to the source ADU flow to which this ADU belongs to. It is assumed that a single byte is sufficient, or said differently, that no more than 256 flows will be protected by a single instance of FECFRAME.

- The following 2 bytes, L[i] (Length), contain the length of this ADU, in network byte order (i.e., big endian). This length is for the ADU itself and does not include the F[i], L[i], or Pad[i] fields.

Then zero padding is added to ADU i (if needed), in field Pad[i], for alignment purposes up to a size of exactly E bytes. The data unit resulting from the ADU i and the F[i], L[i], and Pad[i] fields, is called ADU Information (or ADUI). Each ADUI contributes to exactly one source symbol of the source block.

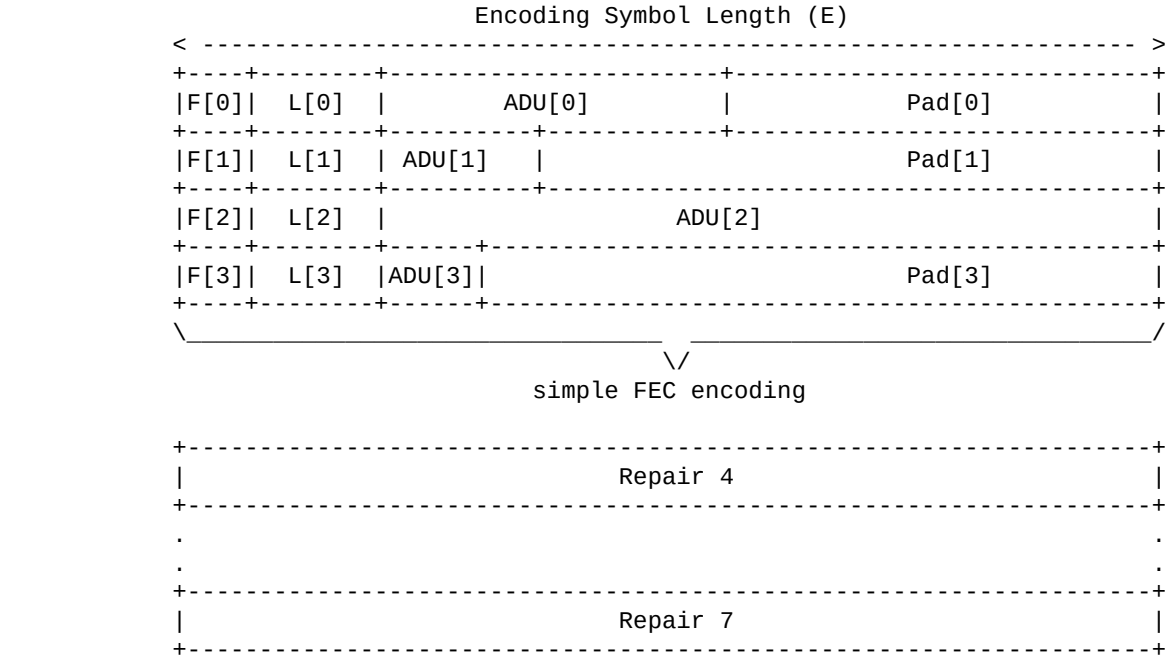

Figure 2: Source block creation, for code rate 1/2 (equal number of source and repair symbols; 4 in this example), and S = 0.

Note that neither the initial 3 bytes nor the optional padding are sent over the network. However, they are considered during FEC encoding. It means that a receiver who lost a certain FEC source packet (e.g., the UDP datagram containing this FEC source packet) will be able to recover the ADUI if FEC decoding succeeds. Thanks to the initial 3 bytes, this receiver will get rid of the padding (if any) and identify the corresponding ADU flow.

5. Simple Reed-Solomon FEC Scheme over GF(2^^m) for Arbitrary ADU Flows

This Fully-Specified FEC Scheme specifies the use of Reed-Solomon codes over GF(2^^m), with 2 <= m <= 16, with a simple FEC encoding for arbitrary packet flows.

5.1. Formats and Codes

5.1.1. FEC Framework Configuration Information

The FEC Framework Configuration Information (or FFCI) includes information that must be communicated between the sender and receiver(s) [RFC6363]. More specifically, it enables the

synchronization of the FECFRAME sender and receiver instances. It includes both mandatory elements and scheme-specific elements, as detailed below.

5.1.1.1. Mandatory Information

- FEC Encoding ID: the value assigned to this Fully-Specified FEC scheme MUST be 8, as assigned by IANA (Section 8).

When SDP is used to communicate the FFCI, this FEC Encoding ID MUST be carried in the 'encoding-id' parameter of the 'fec-repair-flow' attribute specified in RFC 6364 [RFC6364].

5.1.1.2. FEC Scheme-Specific Information

The FEC Scheme-Specific Information (FSSI) includes elements that are specific to the present FEC scheme. More precisely:

- Encoding Symbol Length (E): a non-negative integer, inferior to 2^^16, that indicates either the length of each encoding symbol in bytes ("strict" mode, i.e., if S = 1), or the maximum length of any encoding symbol (i.e., if S = 0).

- Strict (S) flag: when set to 1, this flag indicates that the E parameter is the actual encoding symbol length value for each block of the session (unless otherwise notified by an updated FFCI if this possibility is considered by the use case or CDP). When set to 0, this flag indicates that the E parameter is the maximum encoding symbol length value for each block of the session (unless otherwise notified by an updated FFCI if this possibility is considered by the use case or CDP).

- m parameter (m): an integer that defines the length of the elements in the finite field, in bits. We have: 2 <= m <= 16.

These elements are required both by the sender (Reed-Solomon encoder) and the receiver(s) (Reed-Solomon decoder).

When SDP is used to communicate the FFCI, this FEC scheme-specific information MUST be carried in the 'fssi' parameter of the 'fec-repair-flow' attribute, in textual representation as specified in RFC 6364 [RFC6364]. For instance:

```
a=fec-repair-flow: encoding-id=8; fssi=E:1400,S:0,m:8
```

If another mechanism requires the FSSI to be carried as an opaque octet string (for instance after a Base64 encoding), the encoding format consists of the following 3 octets of Figure 3:

- Encoding symbol length (E): 16-bit field.

- Strict (S) flag: 1-bit field.

- m parameter (m): 7-bit field.

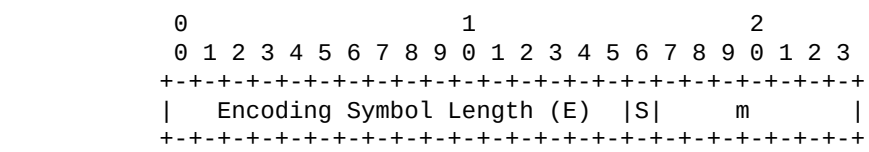

Figure 3: FSSI encoding format.

5.1.2. Explicit Source FEC Payload ID

A FEC source packet MUST contain an Explicit Source FEC Payload ID that is appended to the end of the packet as illustrated in Figure 4.

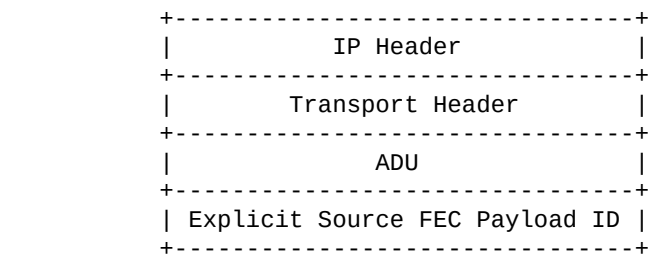

Figure 4: Structure of a FEC Source Packet with the Explicit Source FEC Payload ID.

More precisely, the Explicit Source FEC Payload ID is composed of the Source Block Number, the Encoding Symbol ID, and the Source Block Length. The length of the first 2 fields depends on the m parameter (transmitted separately in the FFCI, Section 5.1.1.2):

- Source Block Number (SBN) ((32-m)-bit field): this field identifies the source block to which this FEC source packet belongs.

- Encoding Symbol ID (ESI) (m-bit field): this field identifies the source symbol contained in this FEC source packet. This value is such that 0 <= ESI <= k - 1 for source symbols.

o Source Block Length (k) (16-bit field): this field provides the number of source symbols for this source block, i.e., the k parameter. If 16 bits are too much when m <= 8, it is needed when 8 < m <= 16. Therefore, we provide a single common format regardless of m.

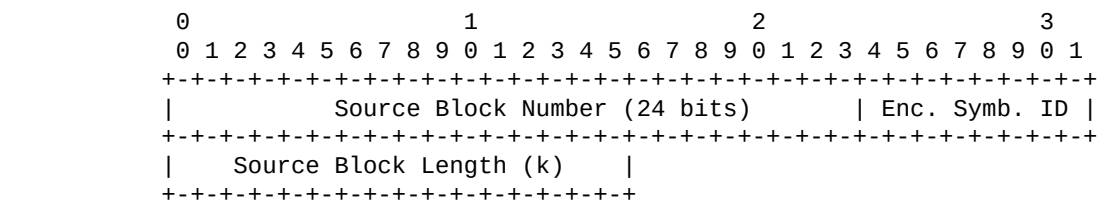

Figure 5: Source FEC Payload ID encoding format for m = 8 (default).

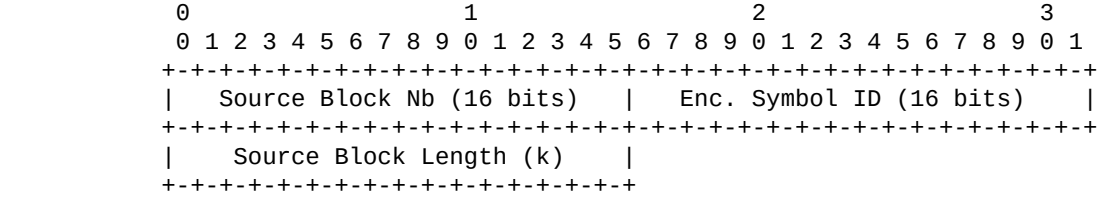

Figure 6: Source FEC Payload ID encoding format for m = 16.

The format of the Source FEC Payload ID for m = 8 and m = 16 are illustrated in Figures 5 and 6, respectively.

5.1.3. Repair FEC Payload ID

A FEC repair packet MUST contain a Repair FEC Payload ID that is prepended to the repair symbol(s) as illustrated in Figure 7. There MUST be a single repair symbol per FEC repair packet.

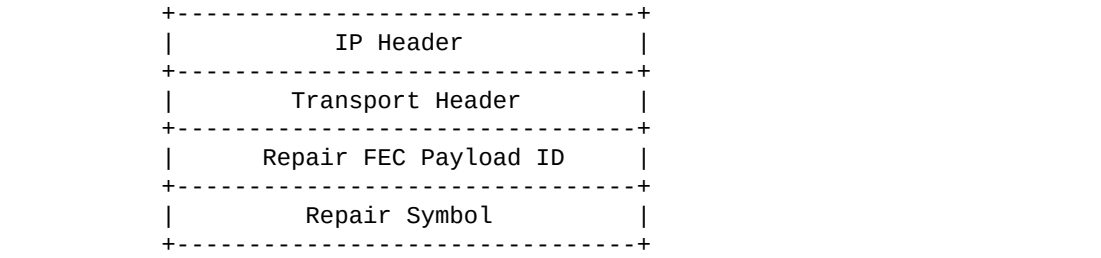

Figure 7: Structure of a FEC Repair Packet with the Repair FEC Payload ID.

More precisely, the Repair FEC Payload ID is composed of the Source Block Number, the Encoding Symbol ID, and the Source Block Length. The length of the first 2 fields depends on the m parameter (transmitted separately in the FFCI, Section 5.1.1.2):

- Source Block Number (SBN) ((32-m)-bit field): this field identifies the source block to which the FEC repair packet belongs.

- Encoding Symbol ID (ESI) (m-bit field): this field identifies the repair symbol contained in this FEC repair packet. This value is such that k <= ESI <= n - 1 for repair symbols.

- Source Block Length (k) (16-bit field): this field provides the number of source symbols for this source block, i.e., the k parameter. If 16 bits are too much when m <= 8, it is needed when 8 < m <= 16. Therefore, we provide a single common format regardless of m.

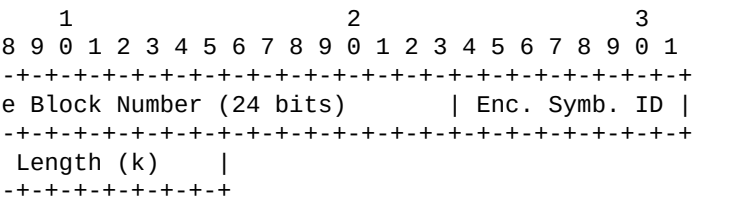

Figure 8: Repair FEC Payload ID encoding format for m = 8 (default).

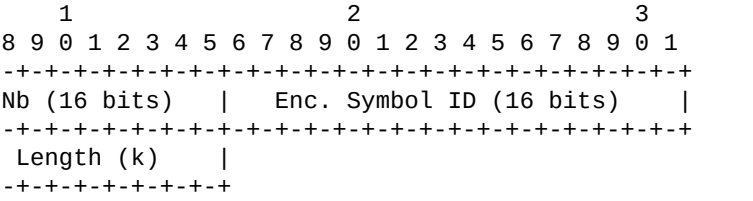

Figure 9: Repair FEC Payload ID encoding format for m = 16.

The format of the Repair FEC Payload ID for m = 8 and m = 16 are illustrated in Figures 8 and 9, respectively.

5.2. Procedures

The following procedures apply:

- The source block creation MUST follow the procedures specified in Section 4.3.
- The SBN value MUST start with value 0 for the first block of the ADU flow and MUST be incremented by 1 for each new source block. Wrapping to zero will happen for long sessions, after value 2^^(32-m) - 1.
- The ESI of encoding symbols MUST start with value 0 for the first symbol and MUST be managed sequentially. The first k values (0 <= ESI <= k - 1) identify source symbols, whereas the last n-k values (k <= ESI <= n - 1) identify repair symbols.
- The FEC repair packet creation MUST follow the procedures specified in Section 5.1.3.

5.3. FEC Code Specification

The present document inherits from Section 8 of [RFC5510], "Reed-Solomon Codes Specification for the Erasure Channel", the specifications of the core Reed-Solomon codes based on Vandermonde matrices.

6. Security Considerations

The FECFRAME document [RFC6363] provides a comprehensive analysis of security considerations applicable to FEC schemes. Therefore, the present section follows the security considerations section of [RFC6363] and only discusses topics that are specific to the use of Reed-Solomon codes.

6.1. Attacks Against the Data Flow

6.1.1. Access to Confidential Content

The Reed-Solomon FEC Scheme specified in this document does not change the recommendations of [RFC6363]. To summarize, if confidentiality is a concern, it is RECOMMENDED that one of the solutions mentioned in [RFC6363] is used with special considerations to the way this solution is applied (e.g., is encryption applied before or after FEC protection, within the end-system or in a middlebox) to the operational constraints (e.g., performing FEC decoding in a protected environment may be complicated or even impossible) and to the threat model.

6.1.2. Content Corruption

The Reed-Solomon FEC Scheme specified in this document does not change the recommendations of [RFC6363]. To summarize, it is RECOMMENDED that one of the solutions mentioned in [RFC6363] is used on both the FEC Source and Repair Packets.

6.2. Attacks Against the FEC Parameters

The FEC Scheme specified in this document defines parameters that can be the basis of several attacks. More specifically, the following parameters of the FFCI may be modified by an attacker (Section 5.1.1.2):

- FEC Encoding ID: changing this parameter leads the receiver to consider a different FEC Scheme, which enables an attacker to create a Denial of Service (DoS).

- Encoding symbol length (E): setting this E parameter to a value smaller than the valid one enables an attacker to create a DoS since the repair symbols and certain source symbols will be larger than E, which is an incoherency for the receiver. Setting this E parameter to a value larger than the valid one has similar impacts when S = 1 since the received repair symbol size will be smaller than expected. On the opposite, it will not lead to any incoherency when S = 0 since the actual symbol length value for the block is determined by the size of any received repair symbol, as long as this value is smaller than E. However, setting this E parameter to a larger value may have impacts on receivers that pre-allocate memory space in advance to store incoming symbols.

- Strict (S) flag: flipping this S flag from 0 to 1 (i.e., E is now considered as a strict value) enables an attacker to mislead the receiver if the actual symbol size varies over different source blocks. Flipping this S flag from 1 to 0 has no major consequences unless the receiver requires to have a fixed E value (e.g., because the receiver pre-allocates memory space).

- m parameter: changing this parameter triggers a DoS since the receiver and sender will consider different codes, and the receiver will interpret the Explicit Source FEC Payload ID and Repair FEC Payload ID differently. Additionally, by increasing this m parameter to a larger value (typically m = 16 rather than 8, when both values are possible in the target use case) will create additional processing load at a receiver if decoding is attempted.

It is therefore RECOMMENDED that security measures are taken to guarantee the FFCI integrity, as specified in [RFC6363]. How to achieve this depends on the way the FFCI is communicated from the sender to the receiver, which is not specified in this document.

Similarly, attacks are possible against the Explicit Source FEC Payload ID and Repair FEC Payload ID: by modifying the Source Block Number (SBN), or the Encoding Symbol ID (ESI), or the Source Block Length (k), an attacker can easily corrupt the block identified by the SBN. Other consequences, that are use case and/or CDP dependent, may also happen. It is therefore RECOMMENDED that security measures are taken to guarantee the FEC Source and Repair Packets as stated in [RFC6363].

6.3. When Several Source Flows Are to Be Protected Together

The Reed-Solomon FEC Scheme specified in this document does not change the recommendations of [RFC6363].

6.4. Baseline Secure FECFRAME Operation

The Reed-Solomon FEC Scheme specified in this document does not change the recommendations of [RFC6363] concerning the use of the IPsec/ESP security protocol as a mandatory to implement (but not mandatory to use) security scheme. This is well suited to situations where the only insecure domain is the one over which FECFRAME operates.

7. Operations and Management Considerations

The FECFRAME document [RFC6363] provides a comprehensive analysis of operations and management considerations applicable to FEC schemes. Therefore, the present section only discusses topics that are specific to the use of Reed-Solomon codes as specified in this document.

7.1. Operational Recommendations: Finite Field Size (m)

The present document requires that m, the length of the elements in the finite field in bits, is such that $2 <= m <= 16$. However, all possibilities are not equally interesting from a practical point of view. It is expected that $m = 8$, the default value, will be mostly used since it offers the possibility to have small to medium sized source blocks and/or a significant number of repair symbols (i.e., $k < n <= 255$). Additionally, elements in the finite field are 8 bits long, which makes read/write memory operations aligned on bytes during encoding and decoding.

An alternative when it is known that only very small source blocks will be used is m = 4 (i.e., k < n <= 15). Elements in the finite field are 4 bits long, so if 2 elements are accessed at a time, read/write memory operations are aligned on bytes during encoding and decoding.

An alternative when very large source blocks are needed is m = 16 (i.e., k < n<= 65535). However, this choice has significant impact on the processing load. For instance, using pre-calculated tables to speed up operations over the finite field (as done with smaller finite fields) may require a prohibitive amount of memory to be used on embedded platforms. Alternative lightweight solutions (e.g., LDPC FEC [RFC5170]) may be preferred in situations where the processing load is an issue and the source block length is large enough [Matsuzono10].

Since several values for the m parameter are possible, the use case SHOULD define which value or values need to be supported. In situations where this is not specified, the default m = 8 value MUST be used.

In any case, any compliant implementation MUST support at least the default m = 8 value.

8. IANA Considerations

Values of FEC Encoding IDs are subject to IANA registration. [RFC6363] defines general guidelines on IANA considerations. In particular, it defines the "FEC Framework (FECFRAME) FEC Encoding IDs" subregistry of the "Reliable Multicast Transport (RMT) FEC Encoding IDs and FEC Instance IDs" registry, whose registration procedure is IETF Review.

This document registers one value in the "FEC Framework (FECFRAME) FEC Encoding IDs" subregistry as follows:

> 8 refers to the Simple Reed-Solomon [RFC5510] FEC Scheme over GF(2^^m) for Arbitrary Packet Flows.

9. Acknowledgments

The authors want to thank Hitoshi Asaeda and Ali Begen for their valuable comments.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, August 2007.

[RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes", RFC 5510, April 2009.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, October 2011.

[RFC6364] Begen, A., "Session Description Protocol Elements for the Forward Error Correction (FEC) Framework", RFC 6364, October 2011.

10.2. Informative References

[Matsuzono10] Matsuzono, K., Detchart, J., Cunche, M., Roca, V., and H. Asaeda, "Performance Analysis of a High-Performance Real-Time Application with Several AL-FEC Schemes", 35th Annual IEEE Conference on Local Computer Networks (LCN 2010), October 2010.

[RFC5053] Luby, M., Shokrollahi, A., Watson, M., and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery", RFC 5053, October 2007.

[RFC5170] Roca, V., Neumann, C., and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes", RFC 5170, June 2008.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, November 2009.

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, April 2010.

[Rizzo97] Rizzo, L., "Effective Erasure Codes for Reliable Computer Communication Protocols", ACM SIGCOMM Computer Communication Review, Vol.27, No.2, pp.24-36, April 1997.

[RS-codec] Rizzo, L., "Reed-Solomon FEC codec (C language)", original codec: http://info.iet.unipi.it/~luigi/vdm98/vdm980702.tgz, improved codec: http://openfec.org/, July 1998.

Authors' Addresses

Vincent Roca
INRIA
655, av. de l'Europe
Inovallee; Montbonnot
ST ISMIER cedex 38334
France

EMail: vincent.roca@inria.fr
URI: http://planete.inrialpes.fr/people/roca/

Mathieu Cunche
INSA-Lyon/INRIA
Laboratoire CITI
6 av. des Arts
Villeurbanne cedex 69621
France

EMail: mathieu.cunche@inria.fr
URI: http://mathieu.cunche.free.fr/

Jerome Lacan
ISAE, Univ. of Toulouse
10 av. Edouard Belin; BP 54032
Toulouse cedex 4 31055
France

EMail: jerome.lacan@isae.fr
URI: http://personnel.isae.fr/jerome-lacan/

Amine Bouabdallah
CDTA
Center for Development of Advanced Technologies
Cite 20 aout 1956, Baba Hassen
Algiers
Algeria

EMail: abouabdallah@cdta.dz

Kazuhisa Matsuzono
Keio University
Graduate School of Media and Governance
5322 Endo
Fujisawa, Kanagawa 252-8520
Japan

EMail: kazuhisa@sfc.wide.ad.jp